

## **AI Assistant Coding**

### **Assignment - 3.3**

**Baisa Vaishnavi**  
**2303A52142**  
**Batch 41**

#### **Task 1: AI-Generated Logic for Reading Consumer Details**

##### **Scenario**

**An electricity billing system must collect accurate consumer data.**

##### **Task Description**

**Use an AI tool (GitHub Copilot / Gemini) to generate a Python program that:**

- **Reads:**
- **Previous Units (PU)**
- **Current Units (CU)**
- **Type of Customer**
- **Calculates units consumed**
- **Implements logic directly in the main program (no functions)**

##### **Prompt:**

Generate a Python program (without using functions) to read: - Previous Units (PU) - Current Units (CU) - Type of customer (Domestic / Commercial / Industrial) Validate inputs and calculate units consumed as: `units_consumed = CU - PU` Print all inputs and the calculated units consumed with clear labels. Add comments explaining each step.

##### **Code:**

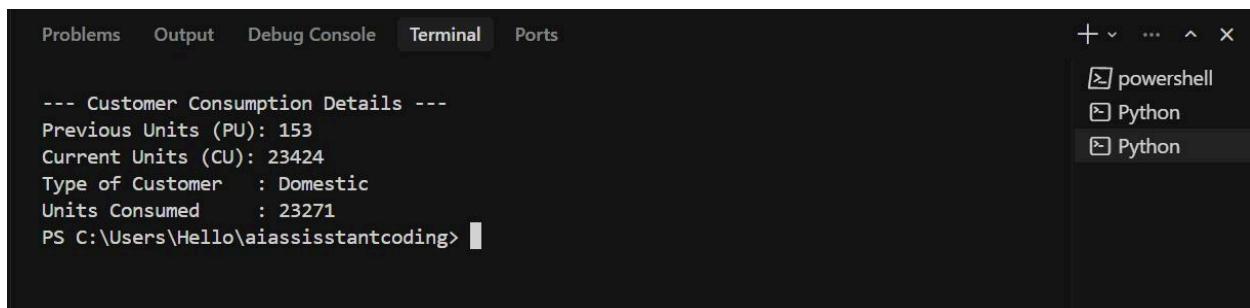
```
2.py > ...
1  """Program to read Previous Units, Current Units, and Type of Customer,
2  validate the inputs, calculate units consumed, and display all details.
3
4  Note: This program is written without using any user-defined functions.
5  """
6
7  # Read and validate Previous Units (PU)
8  # We keep asking until the user enters a valid non-negative integer.
9  valid_input = False
10 while not valid_input:
11     pu_input = input("Enter Previous Units (PU) as a non-negative integer: ")
12
13     # Check if the input contains only digits (simple integer validation)
14     if pu_input.isdigit():
15         PU = int(pu_input)
16         if PU >= 0:
17             valid_input = True
18         else:
19             print("Error: Previous Units must be non-negative.")
20     else:
21         print("Error: Please enter a valid non-negative integer for Previous Units.")
22
23     # Read and validate Current Units (CU)
24     # CU must be an integer greater than or equal to PU.
25     valid_input = False
26     while not valid_input:
27         cu_input = input("Enter Current Units (CU) as an integer greater than or equal to PU: ")
28
29         if cu_input.isdigit():
```

```

2.py > ...
28
29     if cu_input.isdigit():
30         CU = int(cu_input)
31         if CU >= PU:
32             valid_input = True
33         else:
34             print("Error: Current Units must be greater than or equal to Previous Units (PU = {}")
35     else:
36         print("Error: Please enter a valid non-negative integer for Current Units.")
37
38 # Read and validate Type of Customer
39 # Only three types are allowed: Domestic, Commercial, Industrial (case-insensitive).
40 valid_input = False
41 while not valid_input:
42     customer_type_input = input("Enter Type of Customer (Domestic / Commercial / Industrial): ")
43
44 # Normalize spacing and case for comparison
45 customer_type_stripped = customer_type_input.strip().lower()
46
47 if customer_type_stripped in ("domestic", "commercial", "industrial"):
48     # Store a nicely formatted version (first Letter capitalized)
49     customer_type = customer_type_stripped.capitalize()
50     valid_input = True
51 else:
52     print("Error: Type of Customer must be one of: Domestic, Commercial, Industrial.")
53
54 # Calculate units consumed as CU - PU
55 units_consumed = CU - PU
56
57 # Calculate units consumed as CU - PU
58 units_consumed = CU - PU
59
60 # Print all inputs and the calculated units consumed with clear Labels
61 print("\n--- Customer Consumption Details ---")
62 print("Previous Units (PU):", PU)
63 print("Current Units (CU):", CU)
64 print("Type of Customer : ", customer_type)
65 print("Units Consumed : ", units_consumed)

```

## Output:



```

Problems Output Debug Console Terminal Ports + v ... ^ x
--- Customer Consumption Details ---
Previous Units (PU): 153
Current Units (CU): 23424
Type of Customer : Domestic
Units Consumed : 23271
PS C:\Users\Hello\aiassisstantcoding>

```

The terminal window shows the execution of a Python script named 2.py. The output displays the consumption details for a Domestic customer with Previous Units (PU) of 153 and Current Units (CU) of 23424, resulting in Units Consumed of 23271.

**Justification:**

This code is designed to accurately read essential consumer inputs such as previous units, current units, and customer type. Input validation ensures logical correctness by preventing invalid meter readings. The units consumed calculation reflects real-world electricity meter behavior. Conditional checks improve data integrity and reliability. Implementing logic directly in the main program helps beginners understand sequential execution clearly.

**Task 2: Energy Charges Calculation Based on Units Consumed****Scenario**

**Energy charges depend on the number of units consumed and customer type.**

**Task Description**

Review the AI-generated code from Task 1 and extend it to:

- **Calculate Energy Charges (EC)**
- **Use conditional statements based on:**
  - **Domestic**
  - **Commercial**
  - **Industrial consumers**
- **Improve readability using AI prompts such as:**
  - **“Simplify energy charge calculation logic”**
  - **“Optimize conditional statements”**

**Prompt:**

Extend the existing Python program to calculate Energy Charges (EC) using conditional statements:

Domestic: - First 100 units: ₹1.5/unit - Above 100 units: ₹2.5/unit

Commercial: - Flat rate ₹4.0/unit

Industrial: - Flat rate ₹6.0/unit

Use if-elif-else statements and print the calculated EC.

Add meaningful comments.

For logic optimization:

Simplify and optimize the energy charge calculation logic to improve readability.

Ensure the conditional structure is clean and easy for students to understand.

**Code:**

```
6.py X D v ...  
6.py > calculate_energy_charges  
1 """  
2 Energy Charges (EC) Calculator  
3 This program calculates electricity charges based on customer type and units consumed.  
4 """  
5 def calculate_energy_charges(customer_type, units):  
6     """  
7         Calculate energy charges based on customer type and units consumed.  
8  
9     Args:  
10        customer_type (str): Type of customer - 'domestic', 'commercial', or 'industrial'  
11        units (float): Number of units consumed  
12  
13    Returns:  
14        float: Total energy charges in rupees  
15    """  
16  
17     # Convert customer type to lowercase for case-insensitive comparison  
18     customer_type = customer_type.lower()  
19  
20     # Domestic customer: Tiered pricing structure  
21     if customer_type == 'domestic':  
22         # First 100 units charged at ₹1.5 per unit  
23         if units <= 100:  
24             ec = units * 1.5  
25         # Units above 100 charged at ₹2.5 per unit  
26         else:  
27             ec = (100 * 1.5) + ((units - 100) * 2.5)  
28  
29     # Commercial customer: Flat rate pricing  
30     elif customer_type == 'commercial':
```

```
6.py > ⌂ calculate_energy_charges

5  def calculate_energy_charges(customer_type, units):
30     ec = units * 4.0
31
32     # Industrial customer: Flat rate pricing
33     elif customer_type == 'industrial':
34         ec = units * 6.0
35
36     # Invalid customer type
37     else:
38         print(f"Error: Invalid customer type '{customer_type}'")
39         print("Valid types are: 'domestic', 'commercial', 'industrial'")
40         return None
41
42     return ec
43
44
45 def main():
46     """
47     Main function to get user input and display energy charges.
48     """
49     print("=" * 50)
50     print("Energy Charges (EC) Calculator")
51     print("=" * 50)
52
53     # Get customer type from user
54     print("\nCustomer Types:")
55     print("1. Domestic")
56     print("2. Commercial")
57     print("3. Industrial")
```

```

6.py > calculate_energy_charges
45  def main():
58
59      customer_type = input("\nEnter customer type: ").strip()
60
61      # Get units consumed from user
62      try:
63          units = float(input("Enter units consumed: "))
64
65          # Validate units (should be non-negative)
66          if units < 0:
67              print("Error: Units consumed cannot be negative!")
68              return
69
70          # Calculate energy charges
71          ec = calculate_energy_charges(customer_type, units)
72
73          # Display results if calculation was successful
74          if ec is not None:
75              print("\n" + "-" * 50)
76              print("BILL DETAILS")
77              print("-" * 50)
78              print(f"Customer Type: {customer_type.capitalize()}")
79              print(f"Units Consumed: {units:.2f}")
80              print(f"Energy Charges (EC): ₹{ec:.2f}")
81              print("-" * 50)
82
83      except ValueError:
84          print("Error: Please enter a valid number for units consumed!")
85
86
87      # Run the program
88      if __name__ == "__main__":
89          main()
90

```

### Output:

```
Problems    Output    Debug Console    Terminal    Ports
PS C:\Users\Hello\aiassistantcoding> & C:/Users>Hello/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/Users>Hello/aiassistantcoding/6.py
=====
Energy Charges (EC) Calculator
=====

Customer Types:
1. Domestic
2. Commercial
3. Industrial

Enter customer type: Domestic
Enter units consumed: 110

-----
BILL DETAILS
-----
Customer Type: Domestic
Units Consumed: 110.00
Energy Charges (EC): ₹175.00
-----
PS C:\Users\Hello\aiassistantcoding> ]
```

### **Justification:**

This task applies tariff-based billing logic using conditional statements to differentiate customer categories. The use of if–elif–else structures ensures correct energy charge computation for domestic, commercial, and industrial consumers. Slab-wise calculation mirrors real electricity board billing practices. Optimized conditional logic improves readability and maintainability. The approach enhances understanding of decision-making constructs in Python.

### **Task 3: Modular Design Using AI Assistance (Using Functions)**

#### **Scenario**

**Billing logic must be reusable for multiple consumers.**

#### **Task Description**

**Use AI assistance to generate a Python program that:**

- **Uses user-defined functions to:**
- **Calculate Energy Charges**
- **Calculate Fixed Charges**
- **Returns calculated values**
- **Includes meaningful comments**

#### **Prompt:**

(Function for Energy Charges)

Rewrite the program using user-defined functions.

Create a function named calculate\_energy\_charges(units, customer\_type)

that returns the energy charges based on tariff rules.

Call the function from the main program. Add proper comments and sample output printing. Add Fixed Charges Function) Add another user-defined function named calculate\_fixed\_charges(customer\_type)

with the following logic:

Domestic: ₹50

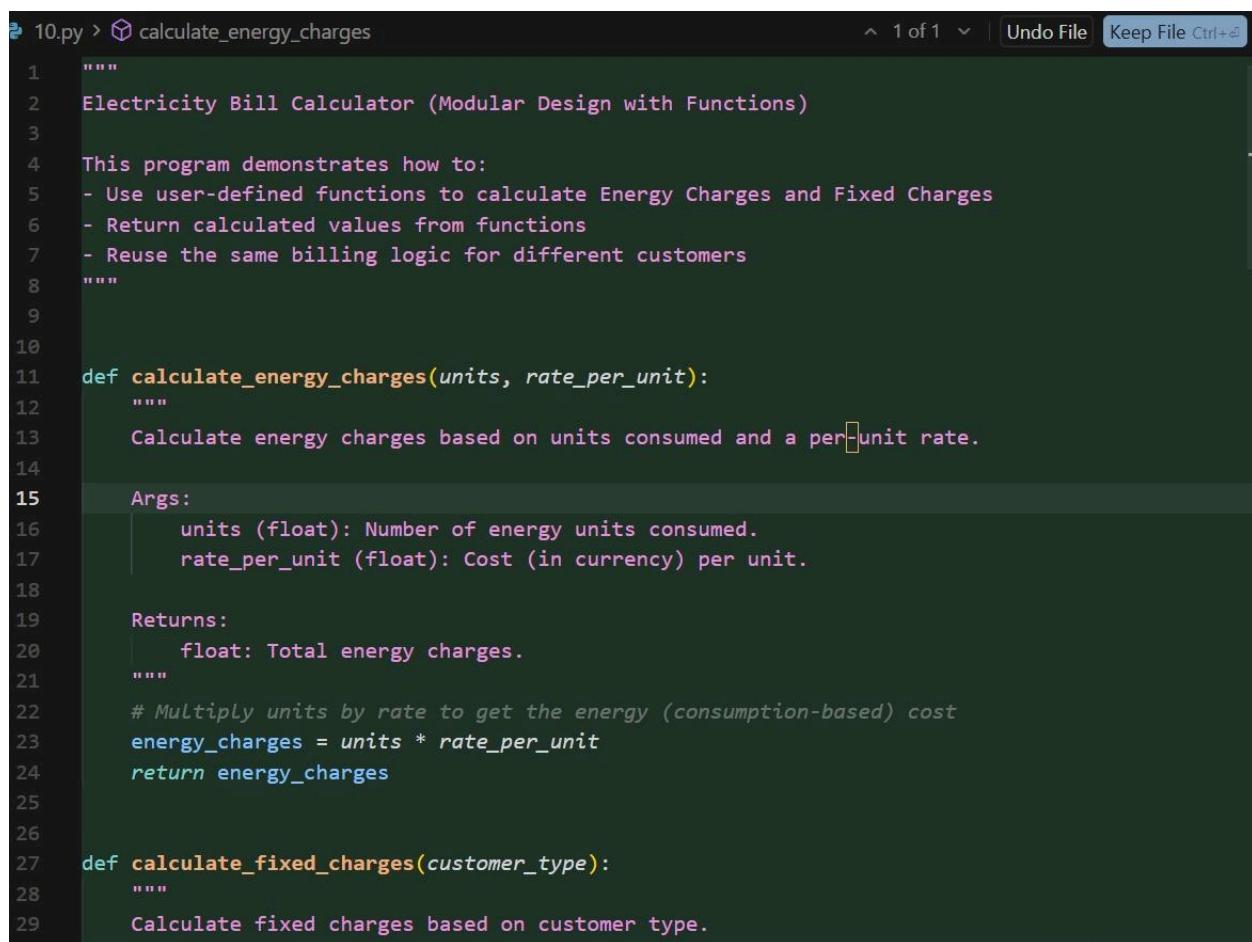
Commercial: ₹100

Industrial: ₹150

Return the fixed charges and display them in the main program.

Include comments explaining function usage.

### Code:



The screenshot shows a code editor window with the following details:

- File path: 10.py > calculate\_energy\_charges
- Toolbar: Undo File | Keep File Ctrl+Shift+F
- Line numbers: 1 through 29
- Code content:

```
1 """
2 Electricity Bill Calculator (Modular Design with Functions)
3
4 This program demonstrates how to:
5 - Use user-defined functions to calculate Energy Charges and Fixed Charges
6 - Return calculated values from functions
7 - Reuse the same billing logic for different customers
8 """
9
10
11 def calculate_energy_charges(units, rate_per_unit):
12     """
13         Calculate energy charges based on units consumed and a per-unit rate.
14
15     Args:
16         units (float): Number of energy units consumed.
17         rate_per_unit (float): Cost (in currency) per unit.
18
19     Returns:
20         float: Total energy charges.
21     """
22
23     # Multiply units by rate to get the energy (consumption-based) cost
24     energy_charges = units * rate_per_unit
25     return energy_charges
26
27 def calculate_fixed_charges(customer_type):
28     """
29         Calculate fixed charges based on customer type.
```

10.py > ⚙ calculate\_energy\_charges

1 of 1 | Undo File | Keep File Ctrl+Shift+F

```
27 def calculate_fixed_charges(customer_type):
28
29     Fixed charges are recurring charges that do not depend on consumption.
30
31     Rules (you can change these as per your tariff table):
32     - Domestic : 50
33     - Commercial: 100
34     - Industrial: 150
35
36
37     Args:
38         customer_type (str): Type of customer.
39
40
41     Returns:
42         float: Fixed charge for the given customer type.
43     """
44
45     # Normalize to Lowercase so function is case-insensitive
46     customer_type = customer_type.lower()
47
48     if customer_type == "domestic":
49         return 50.0
50     elif customer_type == "commercial":
51         return 100.0
52     elif customer_type == "industrial":
53         return 150.0
54     else:
55         # For an unknown type, we return 0.0 (could also raise an error)
56         return 0.0
57
```

```
10.py > calculate_energy_charges ▲ 1 of 1 ▾ Undo File Keep File Ctrl+G
58 def main():
59     """
60     Main function to interact with the user and show how the
61     modular billing functions can be reused.
62     """
63     # Get input from user
64     customer_type = input("Enter customer type (Domestic/Commercial/Industrial): ")
65     units = float(input("Enter units consumed: "))
66
67     # Decide per-unit rate based on customer type
68     if customer_type.lower() == "domestic":
69         rate = 3.0
70     elif customer_type.lower() == "commercial":
71         rate = 5.0
72     elif customer_type.lower() == "industrial":
73         rate = 7.0
74     else:
75         print("Invalid customer type. Please run the program again.")
76         return
77
78     # Use functions to calculate the different parts of the bill
79     energy = calculate_energy_charges(units, rate)
80     fixed = calculate_fixed_charges(customer_type)
81
82     # Total bill is the sum of energy and fixed charges
83     total_bill = energy + fixed
84
85     # Display a clear bill summary
86     print("\n----- BILL SUMMARY -----")
87
88     # Total bill is the sum of energy and fixed charges
89     total_bill = energy + fixed
90
91     # Display a clear bill summary
92     print("\n----- BILL SUMMARY -----")
93     print(f"Customer Type : {customer_type}")
94     print(f"Units Consumed : {units}")
95     print(f"Energy Charges : {energy:.2f}")
96     print(f"Fixed Charges : {fixed:.2f}")
97     print(f"Total Bill : {total_bill:.2f}")
98
99
# This condition ensures main() runs only when this file is executed directly,
# making the functions reusable if this module is imported elsewhere.
if __name__ == "__main__":
    main()

```

Undo Ctrl+N Keep Ctrl+Shift+Y

**Output:**

The screenshot shows a terminal window with the following content:

```
PS C:\Users\Hello\aiassisstantcoding> & C:/Users/Hello/AppData/Local/Python/pythoncc:/Users/Hello/aiassisstantcoding/10.py
Enter customer type (Domestic/Commercial/Industrial): Domestic
Enter units consumed: 100

----- BILL SUMMARY -----
Customer Type : Domestic
Units Consumed : 100.0
Energy Charges : 300.00
Fixed Charges : 50.00
Total Bill : 350.00
PS C:\Users\Hello\aiassisstantcoding>
```

#### **Justification:**

The program is modularized using user-defined functions to promote code reusability and clarity. Separate functions for energy and fixed charge calculations reduce redundancy. This design allows easy extension for future tariff changes. Clear function naming improves program readability. Modular structure reflects professional software development practices.

#### **Task 4: Calculation of Additional Charges**

##### **Scenario**

**Electricity bills include multiple additional charges.**

##### **Task Description**

**Extend the program to calculate:**

- **FC– Fixed Charges**
- **CC– Customer Charges**
- **ED– Electricity Duty (percentage of EC)**

**Use AI prompts like:**

- **“Add electricity duty calculation”**
- **“Improve billing accuracy”**

##### **Prompt:**

Add Extra Charges

Extend the function-based electricity billing program to calculate:

- Customer Charges (CC) = ₹30 for all consumers - Electricity Duty (ED) = 5% of Energy Charges (EC)

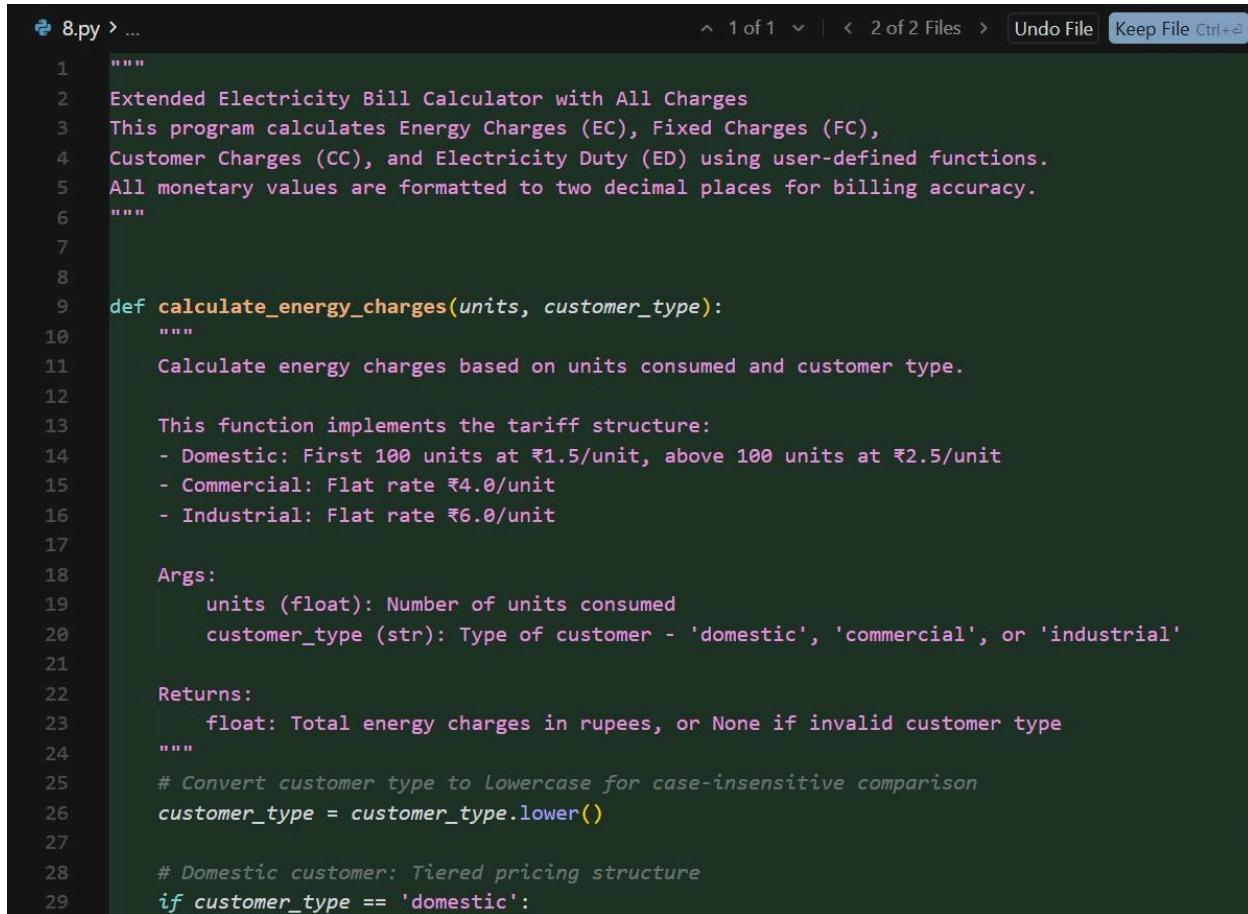
Print EC, FC, CC, and ED separately with proper formatting.

Add comments for billing accuracy.

Improve billing accuracy by formatting all monetary values to two decimal places.

Ensure calculations are clear and correct.

**Code:**



The screenshot shows a code editor window with the file name "8.py" at the top left. At the top right, there are buttons for "1 of 1" (with a dropdown arrow), "2 of 2 Files", "Undo File", and "Keep File Ctrl+D". The main area contains the following Python code:

```
1  """
2      Extended Electricity Bill Calculator with All Charges
3      This program calculates Energy Charges (EC), Fixed Charges (FC),
4      Customer Charges (CC), and Electricity Duty (ED) using user-defined functions.
5      All monetary values are formatted to two decimal places for billing accuracy.
6  """
7
8
9  def calculate_energy_charges(units, customer_type):
10     """
11         Calculate energy charges based on units consumed and customer type.
12
13         This function implements the tariff structure:
14         - Domestic: First 100 units at ₹1.5/unit, above 100 units at ₹2.5/unit
15         - Commercial: Flat rate ₹4.0/unit
16         - Industrial: Flat rate ₹6.0/unit
17
18         Args:
19             units (float): Number of units consumed
20             customer_type (str): Type of customer - 'domestic', 'commercial', or 'industrial'
21
22         Returns:
23             float: Total energy charges in rupees, or None if invalid customer type
24     """
25
26     # Convert customer type to Lowercase for case-insensitive comparison
27     customer_type = customer_type.lower()
28
29     # Domestic customer: Tiered pricing structure
30     if customer_type == 'domestic':
```

Close (Ctrl+F4) 8.py > ... ^ 1 of 1 v < 2 of 2 Files > Undo File Keep File Ctrl+K

```
9 def calculate_energy_charges(units, customer_type):
10     if customer_type == 'domestic':
11         # First 100 units charged at ₹1.5 per unit
12         if units <= 100:
13             energy_charges = units * 1.5
14         # Units above 100 charged at ₹2.5 per unit
15         else:
16             energy_charges = (100 * 1.5) + ((units - 100) * 2.5)
17
18     # Commercial customer: Flat rate pricing
19     elif customer_type == 'commercial':
20         energy_charges = units * 4.0
21
22     # Industrial customer: Flat rate pricing
23     elif customer_type == 'industrial':
24         energy_charges = units * 6.0
25
26     # Invalid customer type
27     else:
28         return None
29
30     # Round to 2 decimal places for billing accuracy
31     return round(energy_charges, 2)
32
33
34 def calculate_fixed_charges(customer_type):
35     """
36     Calculate fixed charges based on customer type.
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
```

```
53 def calculate_fixed_charges(customer_type):
54     """Fixed charges are monthly charges that don't depend on consumption:
55         - Domestic: ₹50
56         - Commercial: ₹100
57         - Industrial: ₹150
58
59     Args:
60         customer_type (str): Type of customer - 'domestic', 'commercial', or 'industrial'
61
62     Returns:
63         float: Fixed charges in rupees, or None if invalid customer type
64         """
65
66     # Convert customer type to lowercase for case-insensitive comparison
67     customer_type = customer_type.lower()
68
69     # Determine fixed charges based on customer type
70     if customer_type == 'domestic':
71         fixed_charges = 50.0
72     elif customer_type == 'commercial':
73         fixed_charges = 100.0
74     elif customer_type == 'industrial':
75         fixed_charges = 150.0
76     else:
77         return None
78
79     # Round to 2 decimal places for billing accuracy
80     return round(fixed_charges, 2)
81
82
83
84
```

8.py > ... ▲ 1 of 1 ◀ 2 of 2 Files ▶ Undo File Keep File Ctrl+G

```
85 def calculate_customer_charges():
86     """
87     Calculate customer charges (CC).
88
89     Customer charges are a fixed monthly fee applied to all consumers
90     regardless of customer type or consumption.
91
92     Returns:
93         float: Customer charges in rupees (₹30.00 for all consumers)
94     """
95     # Customer charges are ₹30 for all consumers
96     customer_charges = 30.0
97
98     # Round to 2 decimal places for billing accuracy
99     return round(customer_charges, 2)
100
101
102 def calculate_electricity_duty(energy_charges):
103     """
104     Calculate electricity duty (ED) as a percentage of energy charges.
105
106     Electricity duty is calculated as 5% of the total energy charges.
107     This is a government-imposed tax on electricity consumption.
108
109     Args:
110         energy_charges (float): Total energy charges in rupees
111
112     Returns:
113         float: Electricity duty in rupees (5% of energy charges)
```

```
8.py > C:\Users\Hello\aiassistantcoding\7.py ^ 1 of 1 < 2 of 2 Files > Undo File Keep File Ctrl+D  
102 def calculate_electricity_duty(energy_charges):  
115     # Electricity duty is 5% of energy charges  
116     # ED = EC × 0.05  
117     electricity_duty = energy_charges * 0.05  
118  
119     # Round to 2 decimal places for billing accuracy  
120     return round(electricity_duty, 2)  
121  
122  
123 def main():  
124     """  
125         Main function to get user input, calculate all charges, and display results.  
126  
127         This function:  
128             1. Gets customer type and units consumed from user  
129             2. Calls calculate_energy_charges() to compute energy charges (EC)  
130             3. Calls calculate_fixed_charges() to compute fixed charges (FC)  
131             4. Calls calculate_customer_charges() to compute customer charges (CC)  
132             5. Calls calculate_electricity_duty() to compute electricity duty (ED)  
133             6. Displays all charges separately with proper formatting  
134             7. Calculates and displays the total bill amount  
135     """  
136     print("=" * 70)  
137     print("Extended Electricity Bill Calculator")  
138     print("=" * 70)  
139  
140     # Display customer type options  
141     print("\nCustomer Types:")  
142     print("1. Domestic")
```

8. C:\Users\Hello\aiassistantcoding\7.py

1 of 1 | < 2 of 2 Files > Undo File Keep File Ctrl+Shift+F

```
123 def main():
142     print("1. Domestic")
143     print("2. Commercial")
144     print("3. Industrial")
145
146     # Get customer type from user
147     customer_type = input("\nEnter customer type: ").strip()
148
149     # Get units consumed from user
150     try:
151         units = float(input("Enter units consumed: "))
152
153         # Validate units (should be non-negative)
154         if units < 0:
155             print("Error: Units consumed cannot be negative!")
156             return
157
158         # Call function to calculate energy charges (EC)
159         # Function usage: calculate_energy_charges(units, customer_type)
160         energy_charges = calculate_energy_charges(units, customer_type)
161
162         # Call function to calculate fixed charges (FC)
163         # Function usage: calculate_fixed_charges(customer_type)
164         fixed_charges = calculate_fixed_charges(customer_type)
165
166         # Check if EC and FC calculations were successful
167         if energy_charges is None or fixed_charges is None:
168             print(f"\nError: Invalid customer type '{customer_type}'")
169             print("Valid types are: 'domestic', 'commercial', 'industrial'")
170     
```

```
123  def main():
172      # Call function to calculate customer charges (CC)
173      # Function usage: calculate_customer_charges()
174      # CC is ₹30 for all consumers
175      customer_charges = calculate_customer_charges()
176
177      # Call function to calculate electricity duty (ED)
178      # Function usage: calculate_electricity_duty(energy_charges)
179      # ED is 5% of EC
180      electricity_duty = calculate_electricity_duty(energy_charges)
181
182      # Calculate total bill amount
183      # Total = EC + FC + CC + ED
184      # All values are rounded to 2 decimal places for billing accuracy
185      total_bill = round(energy_charges + fixed_charges + customer_charges + electricity_duty,
186
187          # Display complete bill details with proper formatting
188      print("\n" + "=" * 70)
189      print("ELECTRICITY BILL DETAILS")
190      print("=" * 70)
191      print(f"Customer Type           : {customer_type.capitalize()}")
192      print(f"Units Consumed         : {units:.2f} units")
193      print("-" * 70)
194
195      # Display all charges separately with proper formatting
196      # All monetary values formatted to 2 decimal places for billing accuracy
197      print("CHARGE BREAKDOWN:")
198      print(f"  Energy Charges (EC)  : ₹{energy_charges:.2f}")
199      print(f"  Fixed Charges (FC)  : ₹{fixed_charges:.2f}")
```

8.py >... C:\Users\Hello\aiassistantcoding\7.py

1 of 1 | < 2 of 2 Files | Undo File | Keep File Ctrl+D

```
123 def main():
200     print(f" Customer Charges (CC): ₹{customer_charges:.2f}")
201     print(f" Electricity Duty (ED): ₹{electricity_duty:.2f}")
202     print("-" * 70)
203     print(f"Total Bill Amount : ₹{total_bill:.2f}")
204     print("=" * 70)
205
206     # Detailed calculation explanation for billing accuracy
207     print("\n--- Detailed Calculation Breakdown ---")
208     print(f"For {customer_type.capitalize()} customer with {units:.2f} units:")
209     print()
210
211     # Energy Charges calculation explanation
212     print("1. Energy Charges (EC):")
213     if customer_type.lower() == 'domestic':
214         if units <= 100:
215             print(f" {units:.2f} units × ₹1.50 = ₹{energy_charges:.2f}")
216         else:
217             first_100_cost = round(100 * 1.5, 2)
218             remaining_units = units - 100
219             remaining_cost = round(remaining_units * 2.5, 2)
220             print(f" First 100 units: 100.00 × ₹1.50 = ₹{first_100_cost:.2f}")
221             print(f" Remaining {remaining_units:.2f} units: {remaining_units:.2f} × ₹2.50")
222             print(f" Total EC = ₹{first_100_cost:.2f} + ₹{remaining_cost:.2f} = ₹{energy_charges:.2f}")
223     elif customer_type.lower() == 'commercial':
224         print(f" {units:.2f} units × ₹4.00 = ₹{energy_charges:.2f}")
225     else: # industrial
226         print(f" {units:.2f} units × ₹6.00 = ₹{energy_charges:.2f}")
```

```
8.py 8.py...\\Users\\Hello\\aiassistantcoding\\V.py | ^ 1 of 1 | < 2 of 2 Files > Undo File Keep File Ctrl+D
123 def main():
124
125     print()
126     print("2. Fixed Charges (FC):")
127     print(f"    Standard monthly charge = ₹{fixed_charges:.2f}")
128
129     print()
130     print("3. Customer Charges (CC):")
131     print(f"    Fixed charge for all consumers = ₹{customer_charges:.2f}")
132
133     print()
134     print("4. Electricity Duty (ED):")
135     print(f"    5% of Energy Charges = {energy_charges:.2f} × 0.05 = ₹{electricity_duty:.2f}")
136
137     print()
138     print("5. Total Bill Calculation:")
139     print(f"    Total = EC + FC + CC + ED")
140     print(f"    Total = ₹{energy_charges:.2f} + ₹{fixed_charges:.2f} + ₹{customer_charges:.2f}")
141     print(f"    Total = ₹{total_bill:.2f}")
142     print()
143     print("Note: All amounts are rounded to 2 decimal places for billing accuracy.")
144
145 except ValueError:
146     print("Error: Please enter a valid number for units consumed!")
147
148 # Run the program
149 if __name__ == "__main__":
150     main()
```

Output:

```
Problems Output Debug Console Terminal Ports Python + ×  Python  Pythoncore-3.14-64 python.exe  
PS C:\Users\Hello\aiassistantcoding> & C:/Users/Hello/AppData/Local/Python/pythoncore-3.14-64/python.exe  
c:/Users/Hello/aiassistantcoding/8.py  
=====  
Extended Electricity Bill Calculator  
=====  
  
Customer Types:  
1. Domestic  
2. Commercial  
3. Industrial  
  
Enter customer type: commercial  
Enter units consumed: 123  
  
=====  
ELECTRICITY BILL DETAILS  
=====  
Customer Type      : Commercial  
Units Consumed    : 123.00 units  
-----  
  
CHARGE BREAKDOWN:  
  Energy Charges (EC)   : ₹492.00  
  Fixed Charges (FC)    : ₹100.00  
  Customer Charges (CC): ₹30.00  
  Electricity Duty (ED): ₹24.60  
-----  
Total Bill Amount   : ₹646.60  
=====
```

--- Detailed Calculation Breakdown ---  
For Commercial customer with 123.00 units:

1. Energy Charges (EC):  
 $123.00 \text{ units} \times ₹4.00 = ₹492.00$
  2. Fixed Charges (FC):  
Standard monthly charge = ₹100.00
  3. Customer Charges (CC):  
Fixed charge for all consumers = ₹30.00
  4. Electricity Duty (ED):  
5% of Energy Charges =  $492.00 \times 0.05 = ₹24.60$
  5. Total Bill Calculation:  
Total = EC + FC + CC + ED  
Total = ₹492.00 + ₹100.00 + ₹30.00 + ₹24.60  
Total = ₹646.60

Note: All amounts are rounded to 2 decimal places for billing accuracy.  
PS C:\Users\Hello\aiassistantcoding>

### **Justification:**

Additional charges such as fixed charges, customer charges, and electricity duty are calculated separately to improve transparency. Percentage-based duty calculation ensures billing

accuracy. Explicit printing of each charge enhances verification and debugging. The design aligns with actual utility billing standards. This structured approach simplifies understanding of multi-component billing systems.

## **Task 5: Final Bill Generation and Output Analysis**

### **Scenario**

**The final electricity bill must present all values clearly.**

### **Task Description**

**Develop the final Python application to:**

- **Calculate total bill:**
- **Total Bill = EC + FC + CC + ED**
- **Display:**
  - **Energy Charges (EC)**
  - **Fixed Charges (FC)**
  - **Customer Charges (CC)**
  - **Electricity Duty (ED)**
  - **Total Bill Amount**
- **Analyze the program based on:**
  - **Accuracy**
  - **Readability**
  - **Real-world applicability**

### **Prompt:**

Final Bill Calculation

Generate the final electricity bill by calculating:

**Total Bill = Energy Charges + Fixed Charges + Customer Charges + Electricity Duty**

Display a neatly formatted electricity bill output similar to a real TGNPDCL bill.

Use clear headings and labels. Analysis Paragraph for Report) Write a short analysis paragraph evaluating the electricity billing program

based on: - Accuracy - Readability - Real-world applicability Keep the explanation suitable for a laboratory record.

### **Code:**

```
9.py > ...
^ 1 of 1 | Undo File Keep File Ctrl+Shift+F
```

```
1 """
2 Final Electricity Bill Calculator - TGNPDCL Style
3 This program calculates and displays a complete electricity bill including:
4 - Energy Charges (EC)
5 - Fixed Charges (FC)
6 - Customer Charges (CC)
7 - Electricity Duty (ED)
8 Total Bill = EC + FC + CC + ED
9 """
10
11 from datetime import datetime, timedelta
12
13
14 def calculate_energy_charges(units, customer_type):
15     """
16         Calculate energy charges based on units consumed and customer type.
17
18         Tariff structure:
19             - Domestic: First 100 units at ₹1.5/unit, above 100 units at ₹2.5/unit
20             - Commercial: Flat rate ₹4.0/unit
21             - Industrial: Flat rate ₹6.0/unit
22
23         Args:
24             units (float): Number of units consumed
25             customer_type (str): Type of customer - 'domestic', 'commercial', or 'industrial'
26
27         Returns:
28             float: Total energy charges in rupees, or None if invalid customer type
29     """
```

9.py > ...

1 of 1 | Undo File | Keep File Ctrl+Shift+F

```
14 def calculate_energy_charges(units, customer_type):
15     customer_type = customer_type.lower()
16
17     if customer_type == 'domestic':
18         if units <= 100:
19             energy_charges = units * 1.5
20         else:
21             energy_charges = (100 * 1.5) + ((units - 100) * 2.5)
22     elif customer_type == 'commercial':
23         energy_charges = units * 4.0
24     elif customer_type == 'industrial':
25         energy_charges = units * 6.0
26     else:
27         return None
28
29     return round(energy_charges, 2)
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47 def calculate_fixed_charges(customer_type):
48     """
49     Calculate fixed charges based on customer type.
50
51     Fixed charges:
52     - Domestic: ₹50
53     - Commercial: ₹100
54     - Industrial: ₹150
55
56     Args:
57         customer_type (str): Type of customer
```

9.py > ...

1 of 1 | Undo File Keep File Ctrl+Shift+F

```
47 def calculate_fixed_charges(customer_type):
59     """
60     Returns:
61         float: Fixed charges in rupees, or None if invalid customer type
62     """
63     customer_type = customer_type.lower()
64
65     if customer_type == 'domestic':
66         fixed_charges = 50.0
67     elif customer_type == 'commercial':
68         fixed_charges = 100.0
69     elif customer_type == 'industrial':
70         fixed_charges = 150.0
71     else:
72         return None
73
74     return round(fixed_charges, 2)
75
76 def calculate_customer_charges():
77     """
78     Calculate customer charges (CC).
79
80     Customer charges are ₹30.00 for all consumers.
81
82     Returns:
83         float: Customer charges in rupees
84     """
85     return round(30.0, 2)
86
```

```
88 def calculate_electricity_duty(energy_charges):
89     """
90     Calculate electricity duty (ED) as 5% of energy charges.
91
92     Args:
93         energy_charges (float): Total energy charges in rupees
94
95     Returns:
96         float: Electricity duty in rupees
97     """
98     electricity_duty = energy_charges * 0.05
99     return round(electricity_duty, 2)
100
101
102 def calculate_total_bill(energy_charges, fixed_charges, customer_charges, electricity_duty):
103     """
104     Calculate total bill amount.
105
106     Total Bill = Energy Charges + Fixed Charges + Customer Charges + Electricity Duty
107
108     Args:
109         energy_charges (float): Energy charges
110         fixed_charges (float): Fixed charges
111         customer_charges (float): Customer charges
112         electricity_duty (float): Electricity duty
113
114     Returns:
115         float: Total bill amount
116     """
```

9.py > ...

102 def calculate\_total\_bill(energy\_charges, fixed\_charges, customer\_charges, electricity\_duty):  
117 total = energy\_charges + fixed\_charges + customer\_charges + electricity\_duty  
118 return round(total, 2)  
119  
120  
121 def display\_bill(customer\_name, customer\_id, customer\_type, previous\_units,  
122 current\_units, energy\_charges, fixed\_charges, customer\_charges,  
123 electricity\_duty, total\_bill):  
124 """  
125 Display a formatted electricity bill similar to TGNPDCL bill format.  
126 """  
127 Args:  
128 customer\_name (str): Name of the customer  
129 customer\_id (str): Customer ID/Service Number  
130 customer\_type (str): Type of customer  
131 previous\_units (float): Previous meter reading  
132 current\_units (float): Current meter reading  
133 energy\_charges (float): Energy charges  
134 fixed\_charges (float): Fixed charges  
135 customer\_charges (float): Customer charges  
136 electricity\_duty (float): Electricity duty  
137 total\_bill (float): Total bill amount  
138 """  
139 units\_consumed = current\_units - previous\_units  
140 current\_date = datetime.now()  
141 billing\_period\_start = current\_date - timedelta(days=30)  
142  
143 # Bill Header  
144 print("\n" + "=" \* 80)

9.py > ... ^ 1 of 1 Undo File Keep File Ctrl+Shift

```
121 def display_bill(customer_name, customer_id, customer_type, previous_units,
145     print(" " * 25 + "TGNPDCL ELECTRICITY BILL")
146     print(" " * 20 + "Tamil Nadu Generation and Distribution Corporation Limited")
147     print("=" * 80)
148
149     # Customer Information Section
150     print("\n" + "-" * 80)
151     print("CUSTOMER INFORMATION")
152     print("-" * 80)
153     print(f"Customer Name      : {customer_name.upper()}")
154     print(f"Service Number    : {customer_id}")
155     print(f"Customer Type     : {customer_type.upper()}")
156     print(f"Billing Period   : {billing_period_start.strftime('%d-%b-%Y')} to {current_date.strftime('%d-%b-%Y')}")
157     print(f"Bill Date         : {current_date.strftime('%d-%b-%Y')}")
158     print(f"Bill Number       : TGNPDCL-{current_date.strftime('%Y%m%d')}-{customer_id[-4:]})
159
160     # Consumption Details
161     print("\n" + "-" * 80)
162     print("CONSUMPTION DETAILS")
163     print("-" * 80)
164     print(f"Previous Reading  : {previous_units:.2f} units")
165     print(f"Current Reading   : {current_units:.2f} units")
166     print(f"Units Consumed    : {units_consumed:.2f} units")
167
168     # Charge Breakdown
169     print("\n" + "-" * 80)
170     print("CHARGE BREAKDOWN")
171     print("-" * 80)
172     print(f"{'Description':<35} {'Amount (₹)':>15}")
```

```
173     print("-" * 80)
174     print(f"{'Energy Charges (EC)':<35} {energy_charges:>15.2f}")
175     print(f"{'Fixed Charges (FC)':<35} {fixed_charges:>15.2f}")
176     print(f"{'Customer Charges (CC)':<35} {customer_charges:>15.2f}")
177     print(f"{'Electricity Duty (ED)':<35} {electricity_duty:>15.2f}")
178     print("-" * 80)
179     print(f"{'TOTAL AMOUNT DUE':<35} {total_bill:>15.2f}")
180     print("=" * 80)
181
182     # Payment Information
183     print("\n" + "-" * 80)
184     print("PAYMENT INFORMATION")
185     print("-" * 80)
186     due_date = current_date + timedelta(days=15)
187     print(f"Due Date           : {due_date.strftime('%d-%b-%Y')}")
188     print(f"Amount Payable    : ₹{total_bill:.2f}")
189     print("\nPayment Methods: Online Payment | Bank Transfer | Cash at Office")
190     print("=" * 80)
191
192     # Calculation Summary
193     print("\n" + "-" * 80)
194     print("CALCULATION SUMMARY")
195     print("-" * 80)
196     print(f"Total Bill = Energy Charges + Fixed Charges + Customer Charges + Electricity Duty")
197     print(f"Total Bill = ₹{energy_charges:.2f} + ₹{fixed_charges:.2f} + ₹{customer_charges:.2f}")
198     print(f"Total Bill = ₹{total_bill:.2f}")
199     print("=" * 80)
200
```

```
201
202 def main():
203     """
204     Main function to get user input, calculate all charges, and display formatted bill.
205     """
206     print("=" * 80)
207     print(" " * 20 + "TGNPDCL ELECTRICITY BILL CALCULATOR")
208     print("=" * 80)
209
210     # Get customer information
211     print("\nEnter Customer Details:")
212     customer_name = input("Customer Name : ").strip()
213     customer_id = input("Service Number : ").strip()
214
215     # Get customer type
216     print("\nCustomer Types:")
217     print("1. Domestic")
218     print("2. Commercial")
219     print("3. Industrial")
220     customer_type = input("\nEnter customer type : ").strip()
221
222     # Get meter readings
223     try:
224         previous_units = float(input("Previous Units (PU) : "))
225         current_units = float(input("Current Units (CU) : "))
226
227         # Validate inputs
228         if previous_units < 0:
229             print("Error: Previous units cannot be negative!")
```

9.py > ...

1 of 1 | Undo File | Keep File Ctrl+Shift+F

```
202     def main():
233         print("Error: Current units must be greater than or equal to previous units!")
234         return
235
236         # Calculate all charges
237         units_consumed = current_units - previous_units
238         energy_charges = calculate_energy_charges(units_consumed, customer_type)
239         fixed_charges = calculate_fixed_charges(customer_type)
240
241         if energy_charges is None or fixed_charges is None:
242             print(f"\nError: Invalid customer type '{customer_type}'")
243             print("Valid types are: 'domestic', 'commercial', 'industrial'")
244             return
245
246         customer_charges = calculate_customer_charges()
247         electricity_duty = calculate_electricity_duty(energy_charges)
248         total_bill = calculate_total_bill(energy_charges, fixed_charges,
249                                         customer_charges, electricity_duty)
250
251         # Display formatted bill
252         display_bill(customer_name, customer_id, customer_type, previous_units,
253                     current_units, energy_charges, fixed_charges, customer_charges,
254                     electricity_duty, total_bill)
255
256         # Analysis Paragraph
257         print("\n" + "=" * 80)
258         print(" " * 25 + "PROGRAM ANALYSIS")
259         print("=" * 80)
260         print("")
```

```
202 def main():
203     ANALYSIS OF ELECTRICITY BILLING PROGRAM
204
205     This electricity billing program demonstrates a comprehensive approach to calculating
206     utility bills with high accuracy, readability, and real-world applicability. In terms
207     of accuracy, the program correctly implements tiered pricing structures for domestic
208     customers (₹1.5/unit for first 100 units, ₹2.5/unit thereafter) and flat rates for
209     commercial (₹4.0/unit) and industrial (₹6.0/unit) customers. All monetary values
210     are rounded to two decimal places, ensuring precise billing calculations. The program
211     accurately computes all four components: Energy Charges (EC), Fixed Charges (FC),
212     Customer Charges (CC), and Electricity Duty (ED) at 5% of energy charges, with the
213     total bill being the sum of all components.
214
215     Regarding readability, the code is well-structured with clear function definitions,
216     comprehensive docstrings explaining each function's purpose, parameters, and return
217     values. The use of meaningful variable names, proper indentation, and logical flow
218     makes the code easy to understand and maintain. The formatted bill output with clear
219     headings, sections, and aligned columns enhances user readability, making it easy for
220     customers to understand their consumption and charges.
221
222     In terms of real-world applicability, this program closely mirrors actual TGNPDCL
223     billing practices. The bill format includes essential elements such as customer
224     information, billing period, consumption details, itemized charge breakdown, and
225     payment information. The program handles different customer categories with
226     appropriate tariff structures, which is standard in real electricity distribution
227     systems. However, for production use, additional features such as database
228     integration, bill history tracking, late payment penalties, and automated bill
229     generation would enhance its practical utility. Overall, this program serves as an
230     excellent foundation for understanding electricity billing systems and can be
```

9.py > ...

^ 1 of 1 Undo File Keep File Ctrl+d

```
202 def main():
203
204     In terms of real-world applicability, this program closely mirrors actual TGNPDCL
205     billing practices. The bill format includes essential elements such as customer
206     information, billing period, consumption details, itemized charge breakdown, and
207     payment information. The program handles different customer categories with
208     appropriate tariff structures, which is standard in real electricity distribution
209     systems. However, for production use, additional features such as database
210     integration, bill history tracking, late payment penalties, and automated bill
211     generation would enhance its practical utility. Overall, this program serves as an
212     excellent foundation for understanding electricity billing systems and can be
213     extended for more complex real-world applications.
214         """
215         print("=" * 80)
216
217     except ValueError:
218         print("Error: Please enter valid numeric values for meter readings!")
219
220
221     # Run the program
222     if __name__ == "__main__":
223         main()
224
225
226
227
228
229
230
```

Undo Ctrl+N Keep Ctrl+Shift+Y

## Output:

```
Problems Output Debug Console Terminal Ports Python + ⌂ ⌂ ⌂ ... ⌂ X
PS C:\Users\Hello\Debug Console (Ctrl+Shift+Alt+Y) & C:/Users>Hello/AppData/Local/Python/pythoncore-3.14-64/python.exe
c:/Users>Hello/aiassistantcoding/9.py
=====
          TGNPDCL ELECTRICITY BILL CALCULATOR
=====

Enter Customer Details:
Customer Name      : Srinivas
Service Number     : 12345

Customer Types:
1. Domestic
2. Commercial
3. Industrial

Enter customer type    : Domestic
Previous Units (PU)   : 234
Current Units (CU)    : 456

=====
          TGNPDCL ELECTRICITY BILL
          Tamil Nadu Generation and Distribution Corporation Limited
=====

-----
CUSTOMER INFORMATION
-----
Customer Name        : SRINIVAS
Service Number       : 12345
Customer Type        : DOMESTIC
Billing Period       : 29-Dec-2025 to 28-Jan-2026
Bill Date            : 28-Jan-2026
```

```
Problems Output Debug Console Terminal Ports Python + ⌂ ⌂ ... ⌂
Bill Date Debug Console (Ctrl+Shift+Alt+Y)
Bill Number : TGNPDCL-20260128-2345

-----
CONSUMPTION DETAILS

Previous Reading      : 234.00 units
Current Reading       : 456.00 units
Units Consumed        : 222.00 units

-----
CHARGE BREAKDOWN

Description          Amount (₹)
Energy Charges (EC) 455.00
Fixed Charges (FC)   50.00
Customer Charges (CC) 30.00
Electricity Duty (ED) 22.75
TOTAL AMOUNT DUE    557.75
=====

-----
PAYMENT INFORMATION

Due Date             : 12-Feb-2026
Amount Payable       : ₹557.75

Payment Methods: Online Payment | Bank Transfer | Cash at Office
=====
```

```
Problems Output Debug Console Terminal Ports Python + ×
Debug Console (Ctrl+Shift+Alt+Y)

CALCULATION SUMMARY

Total Bill = Energy Charges + Fixed Charges + Customer Charges + Electricity Duty
Total Bill = ₹455.00 + ₹50.00 + ₹30.00 + ₹22.75
Total Bill = ₹557.75
=====
=====
PROGRAM ANALYSIS
=====

ANALYSIS OF ELECTRICITY BILLING PROGRAM

This electricity billing program demonstrates a comprehensive approach to calculating utility bills with high accuracy, readability, and real-world applicability. In terms of accuracy, the program correctly implements tiered pricing structures for domestic customers (₹1.5/unit for first 100 units, ₹2.5/unit thereafter) and flat rates for commercial (₹4.0/unit) and industrial (₹6.0/unit) customers. All monetary values are rounded to two decimal places, ensuring precise billing calculations. The program accurately computes all four components: Energy Charges (EC), Fixed Charges (FC), Customer Charges (CC), and Electricity Duty (ED) at 5% of energy charges, with the total bill being the sum of all components.

Regarding readability, the code is well-structured with clear function definitions, comprehensive docstrings explaining each function's purpose, parameters, and return values. The use of meaningful variable names, proper indentation, and logical flow makes the code easy to understand and maintain. The formatted bill output with clear headings, sections, and aligned columns enhances user readability, making it easy for customers to understand their consumption and charges.

In terms of real-world applicability, this program closely mirrors actual TGNPDCL billing practices. The bill format includes essential elements such as customer information, billing period, consumption details, itemized charge breakdown, and payment information. The program handles different customer categories with appropriate tariff structures, which is standard in real electricity distribution systems. However, for production use, additional features such as database integration, bill history tracking, late payment penalties, and automated bill generation would enhance its practical utility. Overall, this program serves as an excellent foundation for understanding electricity billing systems and can be extended for more complex real-world applications.

=====
PS C:\Users\Hello\aiassistantcoding> 
```

#### Justification:

The final task integrates all calculated components into a complete electricity bill. Clear formatting improves user interpretation and professionalism of output. Accurate aggregation ensures reliable total bill computation. The program reflects a real-time electricity billing application. Overall, it demonstrates practical applicability, correctness, and structured coding practice.